(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
5 July 2001 (05.07.2001)

PCT

(10) International Publication Number
WO 01/48634 A2

(51) International Patent Classification[7]: G06F 17/30, 1/00

(21) International Application Number: PCT/CA00/01568

(22) International Filing Date:
21 December 2000 (21.12.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/472,609 27 December 1999 (27.12.1999) US

(71) Applicant: TEXAR SOFTWARE CORP. [CA/CA]; 1101 Prince of Wales Drive, Ottawa, Ontario K2C 3W7 (CA).

(72) Inventors: BACIC, Eugen; 56 Castlethorpe Crescent, Nepean, Ontario K2G 5R1 (CA). WHITE, Tony; 325 Ferndale Avenue, Ottawa, Ontario K1H 6P9 (CA).

(74) Agent: MITCHELL, Richard, J.; Marks & Clerk, P.O. Box 957, Station B, Ottawa, Ontario K1P 5S7 (CA).

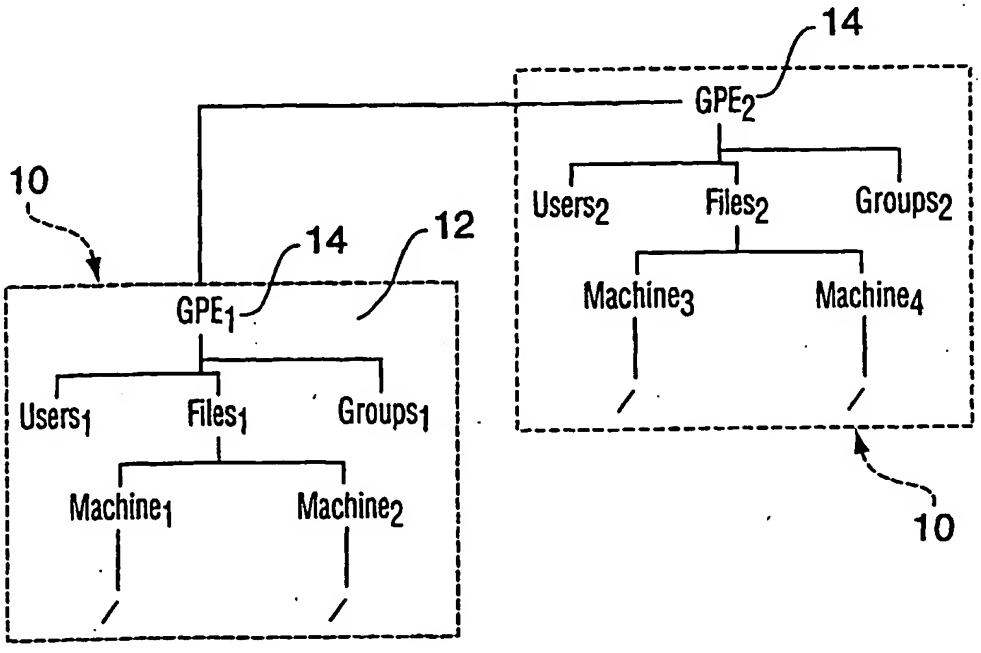
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: VIRTUAL RESOURCE ATTRIBUTE DIRECTORY



(57) Abstract: Computer data is stored in a real file system. Attributes pertaining to the files in the real file system are stored at corresponding locations in the virtual file system, thereby decoupling the storage of attribute information from the data. Typically the file attributes relate to security information.

WO 01/48634 A2

Virtual Resource Attribute Directory

Field of the Invention

This invention relates to computer security, and in particular a method of controlling access to files in a computer system.

Background of the Invention

Computer operating systems, such as Unix, MS DOS and Windows, typically organize files in a tree structure. These files are given attributes, which are stored along with the files in the directory structure. Such attributes can include security controls determining who is permitted to access the files.

The tight binding of security attributes with the information that they secure found in traditional operating systems leads to a restrictive and inflexible security policy implementation that varies from operating system to operating system. As a result, especially in networks running multiple operating systems, this inflexibility makes it difficult to permit central administration of security policy within a system.

Summary of the Invention

According to the present invention there is provided a method of controlling access to computer data, comprising the steps of:

creating a real file system in a computer for storing said data;

creating a virtual file system that mirrors said real file system but lacks the stored data; and

storing attributes pertaining to the files in said file system at corresponding locations in said virtual file system.

Typically the attributes contain security information determining who is permitted access to the files. The virtual file system is known as a virtual resource attribute directory.

The essence of the invention is that it abstracts security away from the simple, fixed attributes that are available within particular operating systems.. The invention ensures that enterprise security policies are defined outside of the operating system, are administered centrally and applied to a single type of structure, the entity. This uniformity ensures policy coherence within an enterprise.

In another aspect the invention provides a virtual resource attribute directory comprising a shadow directory structure mirroring a real file structure and storing attributes of files in said real file structure without the associated data.

The Virtual Resource Attribute Directory (VRAD) defines the structure of the virtualized elements of the information being protected. The principal function of the VRAD is to mediate access to information elements. The VRAD provides a mechanism to ensure that the security attributes required for proper functioning of a security system exist and are accessible. The VRAD is unique for a variety of reasons:

- Non-intrusive to the virtualized system
- Full mapping of extant security controls to security attributes
- Additional security attributes per entity protected for fully realizable security policies
- Portable, non-system dependent
- Extensible and user configurable
- Easily manipulatable

The Virtual Resource Attribute Directory manages the security of information elements stored within it. The VRAD is thus a shadow of the real file system. For example, if the file system is a UNIX file system, then the VRAD would be a virtualization of the UNIX file system. At no point are the actual files modified in any way. No information is stored on the virtualized system other than that associated with the operational agents. There is a clear separation of security and information in a VRAD-managed system. The importance of the security features built into the operating system is significantly diminished.

Brief Description of the Drawings

The invention will now be described in more detail, by way of example only, with reference to the accompanying drawings, in which:-

Figure 1 is an example showing linked security servers; and

Figure 2 is an example showing cross linking of VRAD file systems.

Detailed Description of the Invention

Referring to Figure 1, it will be seen that the Virtual Resource Attribute Directory (VRAD) 10, typically stored on a hard disk, resembles a rooted tree structure 12. This tree structure 12 represents the parent-child relationships that are found in the directory structures of all important file systems. The root 14 of a VRAD can be a security server also known as a generic policy engine, which controls all aspects of security on a network. All elements in the VRAD are represented by entities and proxy entities.

All the VRADs 10 are connected by a super-tree which has at its terminals the VRADs of the virtualized systems as shown in Figure 2.

The various VRADs need not be from the same type of operating system. The VRAD is utilized to create a homogenous representation of all the information that resides within a security controlled realm. This includes unified user and group lists to assist in single sign-on and Authentication Server services.

There remains, at all times, a one-to-one mapping between the physical machine with the resources being protected and a VRAD with the associated security attributes. The two are updated synchronously, via the use of agents, a security server, and message protocol to ensure that each remains perfectly synchronized.

The VRAD 10 stores entities. An entity is the data structure that forms the starting point for all security-related activities. As such, it describes a minimal set of properties that are considered essential for effective security while being fully extensible. Every entity in a VRAD has a unique key generated without relation to the information that it represents; i.e., nothing concerning the data can be inferred from a knowledge of the information and vice versa. The unique key associated with an entity is called the entity identifier, or eid. The eid is represented using a number of bits, n, making the maximum size of the realm 2^n entities. The entity has a security policy associated with it, the security policy being represented by a name in order that policies may be shared by multiple entities in the VRAD. The actual policy is stored in a private part of the VRAD that may only be accessed by security officers.

The attributes that are part of the entity are name, owner, data type, creation timestamp, last modified timestamp and last access timestamp and security policy. The data type

attribute points at a data structure that stores attributes particular to the name of the resource that the entity represents. For example, an entity representing a machine would have the data type machine-ID. A machine-ID instance would store the location of the machine, its IP address, and operating system type.

Another type of data structure stored in a VRAD is a proxy entity. This provides a reference to an entity or another proxy entity that is managed outside of the realm in which the proxy is defined. The function of a proxy entity is to allow a security server to have access to entities outside of the realm without being responsible for their management and to remove the need for the generation of globally-unique entity identifiers across all realms within the enterprise. A proxy entity has a unique key (eid) similar to an entity and a URL that stores the location of the VRAD where the actual entity is stored. The URL consists of two pieces of information. First, the protocol, host and port for a remote security server is present. Second, the eid in *the remote* VRAD is present. A proxy entity can be thought of as a "pointer" to the actual entity. It should be noted that eids are unique within the realm, i.e., no two entities, proxy entities or an entity and a proxy entity can have the same eid.

When information on the actual entity is required, the GPE server managing the realm in which the entity is actually stored is contacted and the information retrieved using the InterRealm Security Protocol (IRSP).

All relationships between entities are stored in a single data structure known as the Entity Relationship (ER) data structure. A one-to-one relation between two entities is stored as a single instance of an ER data structure. A one-to-many, many-to-one or many-to-many relationship is represented as several instances of ER data structures. The ER data structure stores the two entities involved in a relationship, the name of the relationship and a qualifying operator. For example, an ER data structure can be used to store the relationships, "A may read B" and "A may not read B." The difference in representation between the examples in the previous sentence in the value of the associated ER operator. Relationship data structures are used in policies associated with entities to respond to requests for access to an entity. The parent-child relationships that define the structure of a VRAD are stored using ER data structures.

A combination of one or more VRADs is called a Realm. A Realm contains all resources being protected, all users allowed access to those resources, all groups with which those users can be associated, and all physical machines (and their addresses) that represent the Realm. A realm defines a default security policy that is used when individual entities do not have a policy defined for them. This policy ensures that requests for access to resources will always be resolved.

Realms may act as containers for other realms managed by other security servers. The enterprise realm is special in that it acts as a container for all other realms in the enterprise. If an entity is stored within a particular realm, its security is managed by that realm.

Each entity stored within the VRAD has additional attributes and relationships to other entities associated with it. These include unique name, Entity ID, mandatory controls, etc. An entity includes a reference to another data structure by name that contains non-security specific information. For example, the physical location of a machine might be stored and used in a mediation function to prevent information legal in one country from being transmitted to a country in which that information is illegal.

Since the structure is tree-like, it is easy to manipulate the structure via security messaging protocol designed to assist in walking a tree-like structure, and performing actions against it. Any tree-traversal algorithms can be utilized to manipulate the information stored within a VRAD.

VRAD trees can be linked across security servers in order to provide a security solution across an enterprise. The proxy entity concept is used to achieve this.

VRAD trees contain only entity or proxy entity data structures. The VRADs for the resources associated with each machine are stored as subtrees within the VRAD for the realm. The root of this tree is always an entity representing the GPE itself. This entity is called the *realm root*. The parent of the realm root is a proxy entity that represents the realm root, which is one level up in the enterprise security hierarchy. In the case of the enterprise realm root, it is its own parent. It is possible to walk to any realm in the enterprise by walking to the parent of the realm root given appropriate security permissions.

When the parent of the realm root is requested, the proxy entity for the parent is retrieved. The IRSP is used to retrieve the eid of the remote entity if the requesting user has permission to do so. Referring to Figure 2, two realms 10_1, 10_2 are represented. Realms 10_1 and 10_2 are managed by GPE_1 and GPE_2 respectively. When an agent walks from $Machine_1$ to $Files_1$, GPE_1 (realm root), then to the parent GPE (realm root of a parent realm), and finally to $Users_2$ on the remote GPE, the eid of $Users_2$ under GPE_2 is returned. This eid is served up to the user and a new proxy created within $realm_1$. Garbage collection of this proxy entity occurs when the user no longer needs to access the remote entity.

While the above example has demonstrated linking of realms through the realm root entity, cross linking of VRADs at other points within the realm is possible. For example, in Figure 2, a child directory of $machine_4$ in $realm_2$ is managed by $realm_1$. A proxy for dir_{22} is maintained in $realm_2$ and a proxy for the root directory of $machine_4$ is stored in $realm_1$. Walking from the root directory of $machine_4$ takes the user to dir_{22} in $realm_1$. Walking to the children of dir_{22} causes proxy entities to be generated in $realm_2$ that are removed when the user tells the system that they may be discarded or when the user logs out from the system.

The invention provides a flexible approach to file security that is consistent across different operating systems.

We claim:

1. A method of controlling access to computer data, comprising the steps of:

creating a real file system for storing said data;

creating a virtual file system that mirrors said real file system but lacks the stored data; and

storing attributes pertaining to the files in said file system at corresponding locations in said virtual file system.

2. A method as claimed in claim 1, wherein said attributes are security attributes.

3. A method as claimed in claim 2, wherein said virtual file system manages the security attributes stored within it.

4. A method as claimed in claim 3, wherein said virtual file system mediates access to said computer data in said real file system based on said stored attributes in said virtual file system.

5. A method as claimed in claim 4, wherein said virtual file system is organized in a tree structure representing parent-child relationships found in said real file system.

6. A method as claimed in claim 5, wherein said attributes are stored as entities describing the security properties of a corresponding file in said real file structure.

7. A method as claimed in claim 6, wherein each said entity has a unique key generated without relation to the data whose attributes it describes.

8. A method as claimed in claim 7, wherein each said key has a security policy associated with it to permit policies to be shared by multiple entities within the virtual file system.

9. A method as claimed in claim 7, wherein each said entity stores the following attributes: name, owner, data type, creation timestamp, last modified timestamp, last access timestamp, and security policy.

10. A method as claimed in claim 7, wherein said virtual file system also stores proxy entities referencing an actual entity stored in a different virtual file system to permit access to entities stored in said different virtual file system without requiring said first mentioned file system to be responsible for its management.

11. A method as claimed in claim 7, wherein all relationships between entities are stored in a single entity data structure.

12. A method as claimed in claim 7, wherein a plurality of said virtual file systems are linked through their roots.

13. A method as claimed in claim 7, wherein a plurality of said virtual file systems are cross linked at points on the tree structure.

14. A virtual resource attribute directory comprising a shadow directory structure mirroring a real file structure and storing attributes of files in said real file structure without the associated data.

15. A virtual resource attribute directory as claimed in claim 14, wherein said shadow directory structure is a tree structure.

16. A virtual resource attribute directory as claimed in claim 14, wherein said attributes are stored as entities describing the security properties of a corresponding file in said real file structure.

17. A virtual resource attribute directory as claimed in claim 16, wherein each said entity has a unique key generated without relation to the information whose attributes it describes.

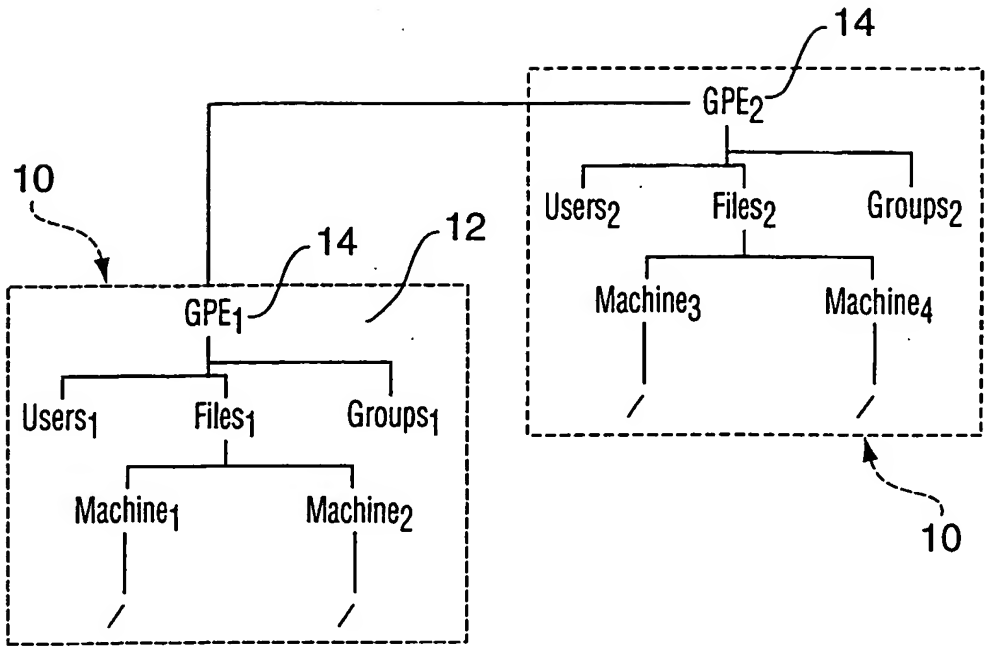

14
GPE2
Users2
Files2
Groups2
Machine3
Machine4
10
14
12
GPE1
Users1
Files1
Groups1
Machine1
Machine2
10

FIG. 1

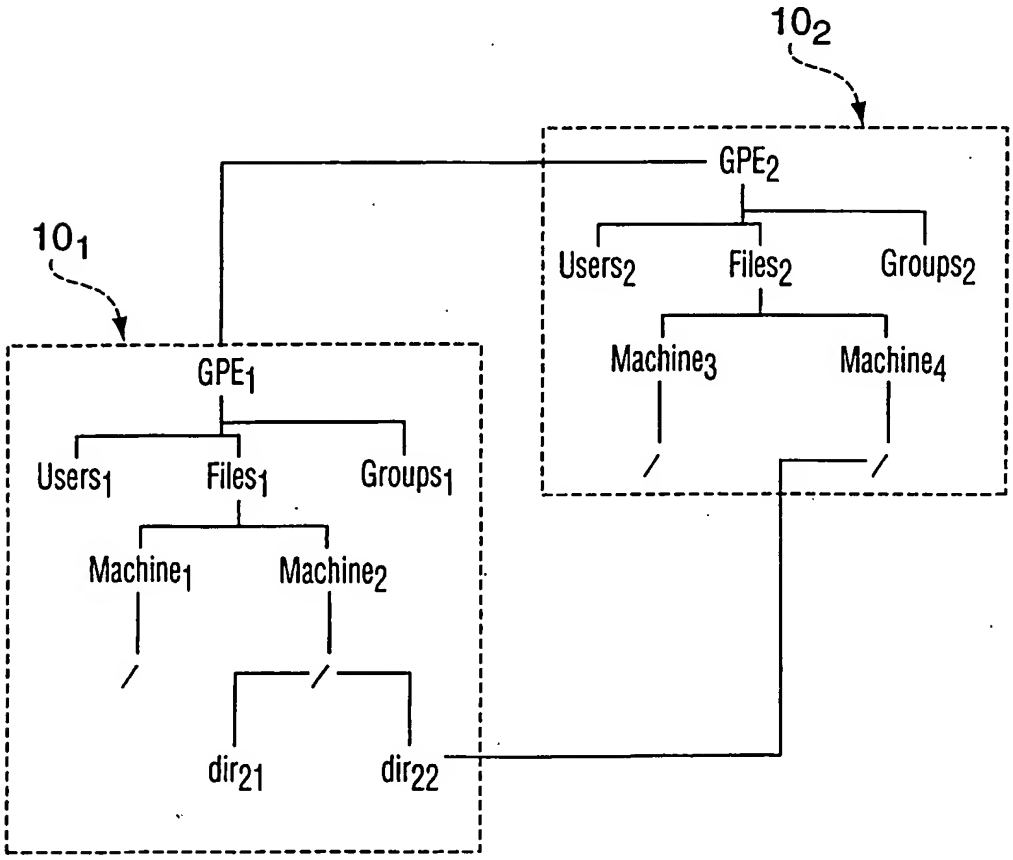

102
GPE2
Users2
Files2
Groups2
Machine3
Machine4
101
GPE1
Users1
Files1
Groups1
Machine1
Machine2
dir21
dir22

FIG. 2